Machine Learning, 47, 35–61, 2002

A New Nonparametric Pairwise Clustering Algorithm Based on Iterative Estimation of Distance Profiles

SHLOMO DUBNOV dubnov@bgumail.bgu.ac.il
Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

RAN EL-YANIV rani@cs.technion.ac.il
Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel

YORAM GDALYAHU yoram@cs.huji.ac.il
School of Computer Science and Engineering and Center for Neural Computation, Hebrew University, Jerusalem 91904, Israel

ELAD SCHNEIDMAN elads@cs.huji.ac.il
School of Computer Science and Engineering, Department of Neurobiology and Center for Neural Computation, Hebrew University, Jerusalem 91904, Israel

NAFTALI TISHBY tishby@cs.huji.ac.il
School of Computer Science and Engineering and Center for Neural Computation, Hebrew University, Jerusalem 91904, Israel

GOLAN YONA golan@cs.cornell.edu
Department of Computer Science, Cornell University, Ithaca, NY 14853-7501, USA

Editor: Douglas Fisher

Abstract. We present a novel pairwise clustering method. Given a proximity matrix of pairwise relations (i.e. pairwise similarity or dissimilarity estimates) between data points, our algorithm extracts the two most prominent clusters in the data set. The algorithm, which is completely nonparametric, iteratively employs a two-step transformation on the proximity matrix. The first step of the transformation represents each point by its relation to *all* other data points, and the second step re-estimates the pairwise distances using a statistically motivated proximity measure on these representations. Using this transformation, the algorithm iteratively partitions the data points, until it finally converges to two clusters. Although the algorithm is simple and intuitive, it generates a complex dynamics of the proximity matrices. Based on this bipartition procedure we devise a hierarchical clustering algorithm, which employs the basic bipartition algorithm in a straightforward divisive manner. The hierarchical clustering algorithm copes with the model validation problem using a general cross-validation approach, which may be combined with various hierarchical clustering methods.

We further present an experimental study of this algorithm. We examine some of the algorithm's properties and performance on some synthetic and 'standard' data sets. The experiments demonstrate the robustness of the algorithm and indicate that it generates a good clustering partition even when the data is noisy or corrupted.

Keywords: nonparametric methods, pairwise distances, hierarchical clustering, Jensen-Shannon divergence, cross-validation, noise robustness

1. Introduction

This paper is concerned with *pairwise clustering*. In this problem a clustering algorithm is given a proximity matrix whose entries correspond to dissimilarity (or similarity) estimates between data points. The goal is to produce a partition of the data, consisting of homogeneous clusters that reveal an inherent structure of the data. This general problem is ill defined and there are many heuristics that try to produce such a partitioning of the data; some are more principled than others.

A few key characteristics separate clustering algorithms into 'families'. The nature of data representation (pairwise relations or feature vectors) is just one of such features of the algorithm that discriminate between clustering methods. Another discrimination is between algorithms that define cost functions and those that rely on well motivated procedural heuristics (e.g. agglomerative methods, see Duda & Hart, 1973; Jain & Dubes, 1988). Data items may be assigned to a unique cluster (hard partition) or to all clusters with some probability (soft partition) according to a membership assignment procedure (Rose, Gurewitz, & Fox, 1992; Buhmann & Hafmann, 1995; Blatt, Wisemann, & Domany, 1996; Pereira, Tishby, & Lee, 1993). When a cost function is defined and the clustering problem is formulated as a discrete optimization problem, it is common to apply iterative optimization techniques in order to approximate a solution. 'Classical' examples are the k-means algorithm (Duda & Hart, 1973), EM algorithms (Dempster, Laird, & Rubin, 1977; Bishop, 1995), CLUSTER/2 (Michalski & Stepp, 1983), AUTOCLASS (Cheeseman et al., 1988) and SNOB (Wallace & Dowe, 1994) among others. Finally, clustering algorithms may provide a hierarchical partition of the data in order to reflect multi scale structure. Hierarchical clustering is associated with the problem of validation, namely how to avoid over-fitting (Linial et al., 1997; Fisher, 1996). An alternative approache to hierarchical clustering include merging clusters using scaling (Wong, 1993).

In this paper we propose a novel algorithm for hierarchical pairwise clustering which is based on a simple transformation of proximity matrices. The iterations of this transformation, starting from the original proximity matrix, converges to a two-valued matrix yielding a bipartition of the data set into two clusters.[1] This bipartition usually reveals the two most prominent clusters in the data set. The hierarchical clustering is obtained by recursively applying this basic bipartition procedure to each of the resulting clusters, represented by their corresponding proximity sub-matrices. The hierarchical algorithm copes with the validation problem using a novel cross-validation method to control each cluster split so that only "meaningful" partitions, which are supported by the data, are allowed.

Although our basic algorithm is quite simple to describe, it performs a global computation that results in a complex transformation of proximity matrices. We cannot supply a general convergence proof of the algorithm at this stage, but based on extensive empirical observations and analytical study of a similar method (Kruskal, 1978; McQuitty & Clark, 1968) we conjecture that the algorithm always converges to a stable data partitioning. Part of this paper presents some of the observations that support our convergence claims. We also

provide some empirical results indicating that the proposed algorithm possesses various attractive properties. Of particular interest are the algorithm robustness to variations in the basic transformation scheme, and the ability to recover the structure even from extremely noisy or corrupted data.

The paper is organized as follows. In Section 2 we define the basic transformation of proximity matrices and study some of its properties. In particular, we demonstrate that the reapplication of this transformation, starting with a proximity matrix, converges to a bipartition of the data set. Section 3 presents our hierarchical clustering algorithm that is based on the bipartition method and a novel cross-validation criterion. In Section 4 we use a synthetic example that demonstrates the algorithm's ability to recover the structure from noisy data under three different noise models. Section 5 presents a preliminary examination of the algorithm's performance with respect to a few data sets, including 'standard' examples from the UCI repository. In Section 6 we discuss the relation of the proposed algorithm to other clustering methods in the literature. Finally, in Section 7 we conclude by summarizing our results, discuss some of their implications and suggest directions for future research.

2. The basic bipartition method

We start with a set $\{1, 2, \ldots, n\}$ of n data points, for which we are given the symmetric proximity matrix $M = (d_{ij})_{i,j=1}^{n}$, where d_{ij} is the pairwise dissimilarity (or similarity) between point i and point j.[2]

Let $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ be an n-dimensional vector in some normed space, and let $\|\mathbf{v}\|$ be the norm of $\mathbf{v}$. For two vectors $\mathbf{u}$ and $\mathbf{v}$, let dist $(\mathbf{u}, \mathbf{v})$ be a proximity measure for $\mathbf{u}$ and $\mathbf{v}$.[3]

We define a two-step transformation of proximity matrices which is the core of our clustering algorithm. Given a proximity matrix $M = (d_{ij})$ the transformation of M to M^{new} is defined via the following two steps.

1. *Normalization step.* For each data point i, let $\mathbf{d}_i = (d_{i1}, d_{i2}, \ldots, d_{in})^t$ be the vector of proximities (distances) of point i to all the other points. (Note that $\mathbf{d}_i$ is the ith column of matrix M). Normalize $\mathbf{d}_i$ by dividing each of its components by $\|\mathbf{d}_i\|$. The resulting normalized vector is thus $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{in}) = (\frac{d_{i1}}{\|\mathbf{d}_i\|}, \frac{d_{i2}}{\|\mathbf{d}_i\|}, \ldots, \frac{d_{in}}{\|\mathbf{d}_i\|})$.
2. *Re-estimation step.* Re-estimate the proximity between every pair of data points i and j, so that $d_{ij}^{new} = \text{dist}(\mathbf{p}_i, \mathbf{p}_j)$.

Thus, the normalization step changes the representation of each data point i so that i is represented by its normalized distances to all other points in the data set. Then, the pairwise distances between all pairs are re-estimated using a vectorial distance measure on their new representations. Intuitively, after the re-estimation step two points are close if they were both close to the same or similar set of points before the transformation (and were both far from the same or similar set of points).

This transformation of M into the new proximity matrix $M^{new} = (d_{ij}^{new})_{i,j=1}^{n}$ is denoted $M^{new} = T(M)$. Let $T^{(k)}(M)$ denote the k-fold composition of T, starting with M. This paper is essentially concerned with studying the properties of the transformation T, and its application for pairwise clustering. We have empirically found that the sequence $T^{(k)}(M)$,

$k = 1, 2, \ldots$, converges fast to a two-valued matrix which is permutation-equivalent to a block-diagonal matrix (with two blocks). In many cases these two blocks correspond to the two most prominent clusters in the data set. This holds for the whole variety of norms and proximity functions we have experimented with, including various L_p norms, various L_p distances and a statistical dissimilarity measure (see Section 2.3).

2.1. The L_1 norm and the Jensen-Shannon divergence

In this section we restrict the transformation T by using the L_1 norm and Jensen-Shannon (JS) divergence (to be defined). Using the L_1 norm in the normalization steps transforms the vector $\mathbf{d}_i$ of distances, representing point i, into $\mathbf{p}_i = (\frac{d_{i1}}{\sum_k d_{ik}}, \frac{d_{i2}}{\sum_k d_{ik}}, \ldots, \frac{d_{in}}{\sum_k d_{ik}})$. This vector of normalized distances may be interpreted as a probability distribution over the set of data points. We therefore turn to a statistical dissimilarity measure in the re-estimation step.

At the outset, a natural candidate for a statistical distance measure between distributions is the Kullback-Leibler (KL) divergence (Kullback, 1959) between $\mathbf{p}_i$ and $\mathbf{p}_j$ defined to be

$$D^{KL}[\mathbf{p}_i \| \mathbf{p}_j] = \sum_k p_{ik} \log_2 \frac{p_{ik}}{p_{jk}}$$

However, the KL-divergence is an asymmetric and unbounded measure and is thus inappropriate here. A more suitable statistical dissimilarity measure in this case is the Jensen-Shannon (JS) divergence (Lin, 1991; Gutman, 1989) (also known as "divergence to the mean") defined as follows. Given two empirical probability distributions (samples) $p(x)$ and $q(x)$, for every $0 \leq \lambda \leq 1$ their λ-JS divergence is defined as

$$D_\lambda^{JS}[p(x) \| q(x)] = \lambda D^{KL}[p(x) \| r(x)] + (1 - \lambda) D^{KL}[q(x) \| r(x)] \tag{1}$$

where

$$r(x) = \lambda p(x) + (1 - \lambda) q(x) \tag{2}$$

can be shown to be the most likely source of both $p(x)$ and $q(x)$ (El-Yaniv, Fine, & Tishby, 1997), with λ as a prior. Without a-priori information about the relative likelihood of p and q we use $\lambda = \frac{1}{2}$. The JS-divergence has various attractive properties. Unlike the KL-divergence, it is a symmetric and more importantly, a bounded measure. The JS-divergence also has a variety of important statistical interpretations. For our purposes, it should be mentioned that its value in bits is proportional to the minus logarithm of the probability that the two empirical distributions represent samples from the same source (El-Yaniv, Fine, & Tishby, 1997; Schreibman, 2000). To summarize, the re-estimation of the distance between point i and point j in this case is

$$\begin{aligned} d_{ij}^{new} &= D_{\frac{1}{2}}^{JS}[\mathbf{p}_i \| \mathbf{p}_j] \\ &= \frac{1}{2}\left(\sum_k p_{ik} \log \frac{p_{ik}}{\frac{1}{2}(p_{ik} + p_{jk})} + \sum_k p_{jk} \log \frac{p_{jk}}{\frac{1}{2}(p_{jk} + p_{ik})} \right) \end{aligned} \tag{3}$$

Reapplying the transformation we get the iterative set of equations,

$$p_{ij}(t+1) = \frac{d_{ij}(t)}{\sum_k d_{ik}(t)} \tag{4}$$

$$\begin{aligned} d_{ij}(t+1) &= D^{JS}[\mathbf{p}_i(t+1) \| \mathbf{p}_j(t+1)] \\ &= \frac{1}{2}\left(\sum_k p_{ik}(t+1) \log \frac{p_{ik}(t+1)}{\frac{1}{2}(p_{ik}(t+1) + p_{jk}(t+1))} \right. \\ &\qquad \left. + \sum_k p_{jk}(t+1) \log \frac{p_{jk}(t+1)}{\frac{1}{2}(p_{jk}(t+1) + p_{ik}(t+1))} \right) \end{aligned} \tag{5}$$

An example of a sequence of reapplication of this transformation is depicted in figure 1. Panel 1(A) depicts 100 random 2D vectors that were generated by two 2D Gaussians. The initial proximity matrix M for these vectors was computed using the Euclidean distances between the 2D vectors (figure 1(B)). The points are arranged so that the first 50 vectors, represented by the indices 1–50, were generated from one of the Gaussians, and the second 50 vectors, represented by the indices 51–100, were generated by the second Gaussian. The gray level value of the entry (i, j) depicts the distance between points i and j, where black is zero distance (as in the case of the entries on the diagonal), and white is the maximal distance in the matrix. The sequence of proximity matrices (Panels C, D, E and F) depict the proximity matrices $T(M)$, $T^{(2)}(M)$, $T^{(5)}(M)$ and $T^{(27)}(M)$, respectively. This sequence converges to a binary matrix[4] that splits the data points into two clusters corresponding to the two Gaussians.

As each data point originates from one of two sources, perfect clustering would be to assign all the points from one source to one cluster, and the points from another source to the other. The distance matrix in this case would be a binary block diagonal matrix (where each block is of size 50). However, in the current example there are 3 classification errors,[5] which we see in the final matrix as 'negative' lines, reflecting that the corresponding point is close to the points that originated from the other source. We note that the number of errors when using the optimal Bayesian classifier is three as well.

2.2. *Possible convergence solutions and fixed point equations*

The algorithm converges when the solutions of Eqs. (4) and (5) are 'self-consistent', i.e. for all i's and j's, $p_{ij}(t+1) = p_{ij}(t)$ and $d_{ij}(t+1) = d_{ij}(t)$. Thus, the possible results of the algorithm are the family of solutions of the set of the following 'steady-state' equations

$$\begin{aligned} \forall i, j \quad d^*_{ij} &= \frac{1}{2} \sum_k \frac{d^*_{ik}}{\sum_l d^*_{il}} \log \frac{\frac{d^*_{ik}}{\sum_l d^*_{il}}}{\frac{1}{2}\left(\frac{d^*_{ik}}{\sum_l d^*_{il}} + \frac{d^*_{jk}}{\sum_l d^*_{jl}} \right)} \\ &\quad + \frac{1}{2} \sum_k \frac{d^*_{jk}}{\sum_l d^*_{jl}} \log \frac{\frac{d^*_{jk}}{\sum_l d^*_{jl}}}{\frac{1}{2}\left(\frac{d^*_{jk}}{\sum_l d^*_{jl}} + \frac{d^*_{ik}}{\sum_l d^*_{il}} \right)} \end{aligned} \tag{6}$$

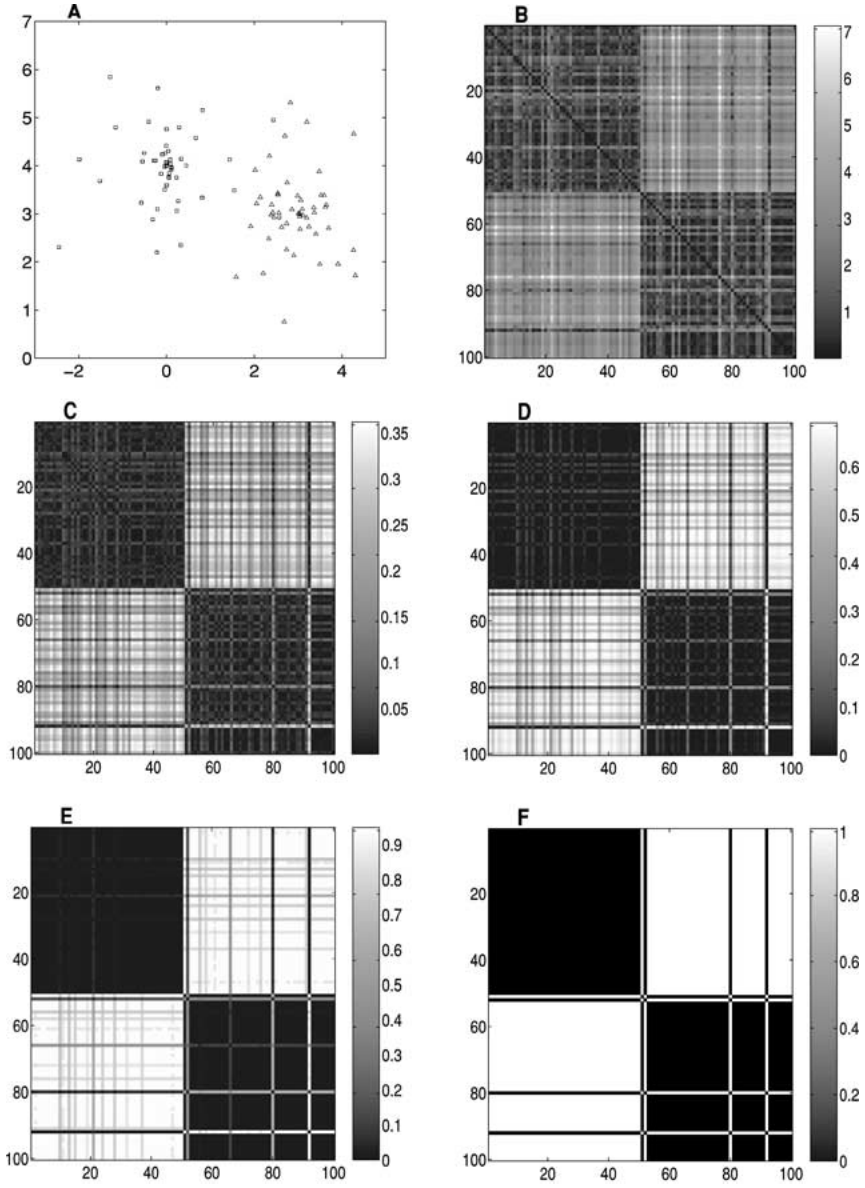

Figure 1. A sequence of transformations applied to data points generated from two $2D$ Gaussians. (A) 100 data points from two Gaussians (fifty points from each). (B) The original matrix of pairwise distances between the data points (Euclidean). The gray-scale bar indicates the gray level code for the values of the matrix. (C) The pairwise proximity matrix resulting from applying T to the matrix presented in B. Due to the nature of JS-divergence, the values are bounded to be between 0 and 1. Here we stretch the gray-scale code such that white corresponds to 0.37 (the maximal value obtained). (D–E) As in C, only for $T^{(2)}$ and $T^{(5)}$ (again, note the change in the gray-scale). (F) The results for $T^{(27)}$, when we obtain convergence to a binary matrix, which can be permutated into a block-diagonal matrix.

So far, we have not been able to get a closed-form solution of this set of equations in general. The main difficulty lies in the nature of coupling between all the d_{ij}'s.[6] However, it is not hard to prove that any binary matrix which is permutation-equivalent to a binary block-diagonal matrix is a fixed-point of T. An example of a member of this family of solutions is the following M^*,

$$M^* = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

We note that this example (or any of the other members of this family of solutions) may not be a stable solution. Performing a general stability analysis of these equations encounters the same problems as in solving the steady-state equations. Kruskal (1978) has shown that for a similar algorithm, which iteratively calculates the correlation between data points based on the correlation matrix columns (McQuitty & Clark, 1968), an equivalent solution is indeed stable, and has calculated a bound on the basin of attraction. We have not been able to extend his analysis to our case.

Extensive empirical investigation of the algorithm suggests that T always converges rather fast[7] (see e.g., Table 1). In almost all the cases T converged to a binary block-diagonal matrix with exactly two blocks. We have encountered very few examples in which T converged to a block-diagonal (non-binary) matrix with three blocks (and these solutions had special symmetry properties).

Table 1. Number of iterations until convergence and number of errors for various norms and proximity measures.

	Norm				
Proximity measure	L_1	L_2	L_3	L_4	L_∞
JS	27, 3	–	–	–	–
L_1	353, 2	25, 3	24, 3	23, 3	23, 3
L_2	230, 2	38, 3	40, 3	42, 3	42, 3
L_3	201, 2	63, 3	106, 2	81, 2	81, 2
L_4	194, 2	129, 3	106, 2	105, 2	104, 2

In each table entry (t, e), t is the number of iterations until ϵ-convergence, with $\epsilon = 1 \times 10^{-4}$, and e is the number of errors (missclassifications with respect to the known sources). In all runs the distance matrix used is the one presented in figure 1(B).

2.3. Other norms and proximity measures

The specific norm one chooses to use in the normalization step (which re-scales the set of pairwise distances for each of the data points) can affect the results of the re-estimation step, and therefore may generate a different transformation. Similarly, in the re-estimation step one can use a variety of vectorial proximity measures such as L_p-based distances, different correlations coefficients and other statistics.

Surprisingly, in our experience, the transformation is not very sensitive to the choice of norm and proximity measure. Specifically, we have experimented with L_p, $p = 1, 2, 3, 4$ and L_∞ norms and distances, and also used a Boltzmann-like normalization $p_{ij} = \frac{exp[-\beta d_{ij}]}{\sum_k exp[-\beta d_{ik}]}$, with different values of β (which is the inverse of a "temperature"). Table 1 summarizes the results of the bipartition algorithm using various norms and proximity measures with respect to the data set of figure 1. These results reflect the strength of our general two-step transformation scheme. For the different norms and proximity measures we have used, the number of errors is either 2 or 3 in all runs with exactly the same two or three errors (and all runs converge within 23 to 230 iterations).

We should note that in principle, one could also use a rescaling of the distances which does not normalize the distances vector to 1. However, if the proximity measure requires some regularity conditions, (e.g., the JS-divergence requires that the vectors of distances are probability vectors) the normalization should accommodate this requirement.

3. Cross-validated pairwise hierarchical clustering

As discussed in Section 2, our basic algorithm converges to a bipartition of the data set. This partition will tend to reveal the coarse structure of the data, namely the two most prominent clusters. A recursive application of the algorithm to each of these clusters results in a natural hierarchical clustering algorithm. Specifically, this algorithm is recursively defined as follows. Let $S = \{1, 2, \ldots, n\}$ be the index set corresponding to n data points and suppose that $M_S = (d_{ij})_{i,j=1}^n$ is the corresponding proximity matrix. An application of the basic algorithm partitions the index set S into two complementary subsets, S_1 and S_2. We now project M_S over S_1 and S_2, respectively, to create two proximity sub-matrices M_{S_1} and M_{S_2}. We continue by recursively applying the algorithm to M_{S_1} and M_{S_2}, etc. The resulting hierarchical clustering computed by this algorithm can be represented by a partition tree (dendrogram) in the standard way (see e.g. Jain & Dubes, 1988).

The inherent nature of the algorithm to split the data into parts, has a 'flipside', in the form of its tendency to find such structure even when it is not highly significant; i.e., given a data set, the hierarchical algorithm will continue partitioning the data until each data point will reside in a single cluster. The problem of avoiding "meaningless" partitions so that the leaves in the resulting hierarchy tree would reflect the 'true' structure of the data is a major challenge in hierarchical clustering, also known as the *validation problem* (see e.g. Jain & Dubes, 1988).

We present a novel general approach for validation in (pairwise) hierarchical clustering (we postpone the comparison to other pruning approaches to the end of this section). At each stage of the hierarchy computation, before applying our bipartition method to a proximity

matrix M_S, we randomly partition the set S of data points into three subsets S_1, S_2 and S_3, such that $|S_1| \approx |S_2| \approx |S_3| \approx n/3$, and $S = S_1 \cup S_2 \cup S_3$. Let $A = S_1 \cup S_2$ and $B = S_2 \cup S_3$; that is, $A \cap B = S_2$. We now apply the clustering algorithm to A and to B and count m, the number of data points in S_2 on which these two independent clustering applications agree. Define $\rho = m/|S_2|$ to be the *cross-validation index*. This index measures the agreement percentage, over $A \cap B$, of the two bipartitions of our clustering algorithm when applied to the subsets A and B, respectively.

Clearly, ρ is a random variable whose distribution depends on the data set and our bipartition method. Intuitively, if there is a structure in the data, it is likely that the two subsets A and B will each capture this structure and therefore will impose similar partitions of $A \cap B$. Consequently, ρ will be large on the average (w.r.t. the distribution of choices of A and B). Conversely, if there is no clear structure in the data, (i.e. the data is 'uniform'), it is unlikely that both A and B will impose the same or similar partition over $A \cap B$ and therefore ρ will be small on the average. In order to get a 'reliable' estimate of ρ, we average the cross validation index over a independent runs; in each run a new random choice of the sets A and B is performed.

Figure 2 shows a data set of 400 points to which we apply the hierarchical algorithm. The top left cluster is sampled from a uniform distribution on a disc and contains 200 points. The bottom two clusters are sampled independently from two uniform discs of a smaller radius and contain 100 points each.

Figure 3 shows the first three levels of the hierarchy tree obtained for this set. The 400 points (see the root of figure 3) are split into two equal size parts (200 points each)–one is indeed the upper left cluster and the other is made up of the two bottom right clusters. The cross-validation index of this split is $\rho = 1.0$, which means perfect agreement between the

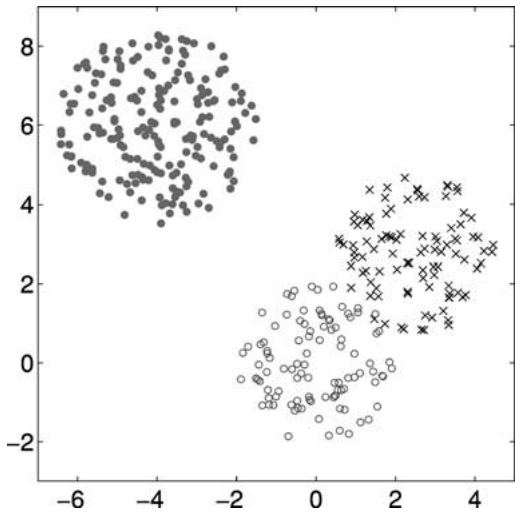

Figure 2. 400 data points sampled from three 2D uniform discs. The top left cluster contains 200 points, and the lower two contain 100 points each.

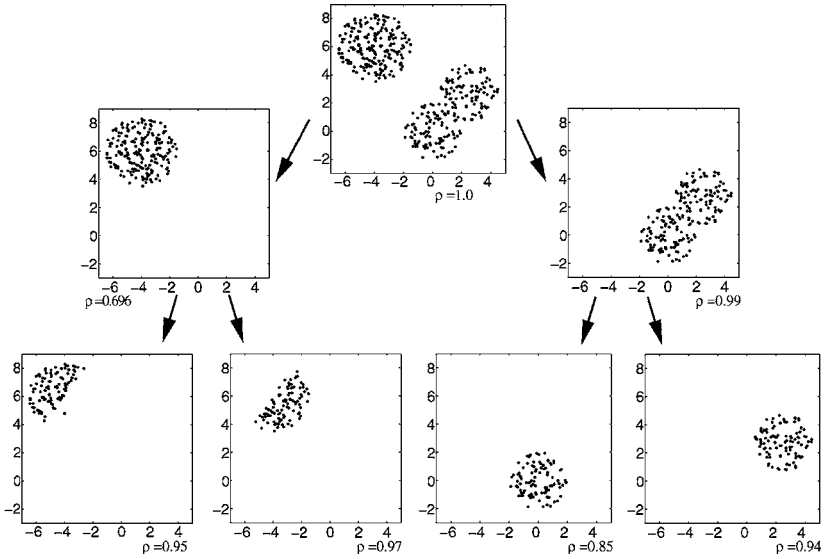

Figure 3. The tree hierarchy representing the clustering result of the data set of figure 2. Each node contains the data points. The cross-validation index of each node's split is denoted below the node.

partitions of the sets A and B in the cross-validation protocol. The two splits in the second level of the tree are very different in nature. The cross-validation index for the right son of the root (the 200 points from the two rightmost discs) is 0.99, implying a "meaningful" partition. The result is a split into the two discs, with one mistake (which is what we would expect from the optimal Bayes classifier as well). The cross-validation value of the split of the left son (the 200 point disc) is smaller ($\rho = 0.696$), implying that this is not necessarily a natural split.[8]

In order to use ρ for deciding whether to allow a split, we should "calibrate" our scale. We use the following "internal" approach (see e.g. Jain & Dubes, 1988), and let the data dictate the scale that determines whether ρ is "unusually" large. When considering a split corresponding to the proximity matrix M_S, we generate a random proximity matrix M_{rand} such that M_{rand} is symmetric and the same dimension as M_S. The elements of M_{rand} are randomly generated so that the off-diagonal elements of M_{rand} have the same first two moments as the off-diagonal elements of M_S. Our goal now is to estimate the distribution of the cross-validation index corresponding to a split of M_{rand}. Lacking a theoretical identification of this distribution we heuristically estimate it using Monte-Carlo sampling; i.e., we apply the cross-validation algorithm to M_{rand} sufficiently many times. Thus, we collect a set $R = \{\rho_i\}_{i=1}^{m}$ of m samples of the cross-validation index. Note that such sampling procedure can be time consuming and therefore the calibration method presented here is not efficient.

Figure 4 shows a random selection of two subsets A and B of the top left cluster of figure 2 (which is the left son in figure 3). The size of each of these sets is approximately 200/3. Set A is marked with filled circles and open ones, set B is marked with crosses and open circles, and hence the cross-validation set, $A \cap B$, is marked with the open circles.

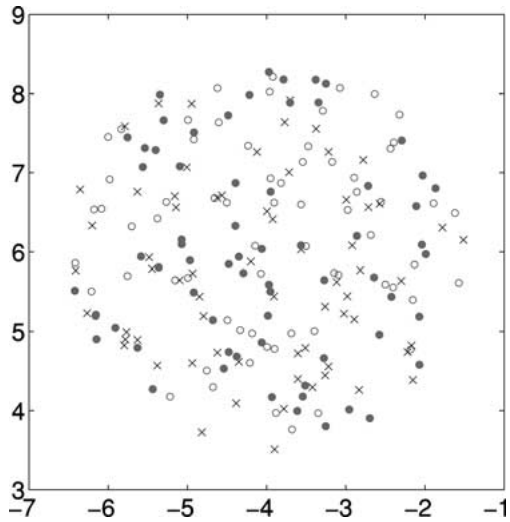

Figure 4. An example of the basics of the cross-validation procedure. The set of points of the left son of figure 3 is randomly divided into the cross-validation sets. Set A (open circles filled ones) and B (crosses and open circles) share the test set $A \cap B$ (open circles). See text for details.

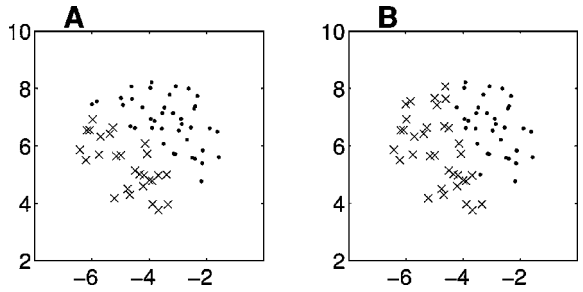

Figure 5. Two different bipartitions of the cross-validation set ($A \cap B$). (A–B) The bipartition of the test set points ($A \cap B$), resulting from two random (different) choices of A and B. The two bipartitions differ by approximately 15%.

Notice that A, B and $A \cap B$, are well mixed and that each of these subsets captures the general structure of the entire cluster.

Figure 5 depicts two random bipartitions of the test set $A \cap B$ corresponding to two random (different) choices of A and B. Since these two bipartitions do not agree on 9 points and the size of the cross-validation set is 67, the cross-validation index of this example is $58/67 \approx 0.865$. (Note that this value is relatively high compared to the average of 0.696 mentioned above.)

Let $\alpha \in (0, 1)$ be a desired significance level (typically $\alpha = 0.95$ or $\alpha = 0.99$) and let P_α be the threshold characterizing the 1-α percentile of set R (i.e., every element in the top 1-α percentile is larger than P_α). If ρ, the cross-validation index we have obtained for the split

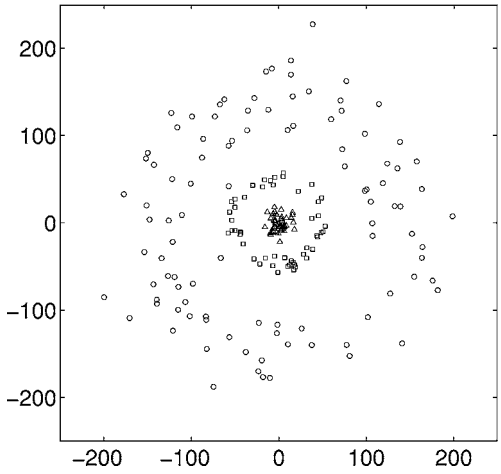

Figure 6. Three concentric rings data. The 50 points of the inner ring were sampled from a ring with a mean radius of 10 units and SD of 5. The middle ring (50 points) had a mean of 50 and a SD of 5 and the outer ring (100 points) had a mean radius of 150 units and an SD of 30 units.

of M_S, is larger than P_α, then we reject, at significance α, the null hypothesis that assumes unstructured (random) data. For the example shown in figure 3, this scheme would result in making the first split with very high confidence level. The hierarchy would stop on the first level of the subtree on the left (i.e., the circle would not be broken into halves), as would be desirable given the true nature of the data. On the right side, the partition would stop at the left leaf (as desired), but would continue even beyond the current level of the tree for the rightmost leaf (which means failure of the scheme for this leaf).

We now turn to an example of data points sampled from concentric rings. This is usually considered to be a harder clustering problem, due to the fact that many algorithms make explicit or implicit assumptions regarding the shape of the clusters. Figure 6 shows 200 points which were sampled from three concentric rings (see the figure caption for details).

Figure 7 shows the results of our hierarchical clustering algorithm on this data set. Also shown are the measured cross-validation indices, and the one percentile threshold $P_{0.99}$, according to which we reject the null hypothesis at significance 0.99. This threshold is computed from 500 random cross-validated splits of the random matrix M_{rand}, as described above.

The first split partitions the entire data set of 200 points into two clusters: the two concentric rings in the middle (93 points), and the outset ring (107 points, 7 of which should have been part of the inner ring). This split has an average cross validation index (with respect to three independent runs) of 0.985 which is larger than the percentile threshold $P_{\alpha=0.99} = 0.78$, and thus we can reject the "no structure" null hypothesis and accept this split. In contrast, the split of the inner cluster of 48 points (left most grandchild of the root) yielded a cross-validation index of 0.81 while $P_{\alpha=0.99} = 0.94$. Here we cannot reject the null hypothesis with high confidence, and therefore we cannot accept this split.

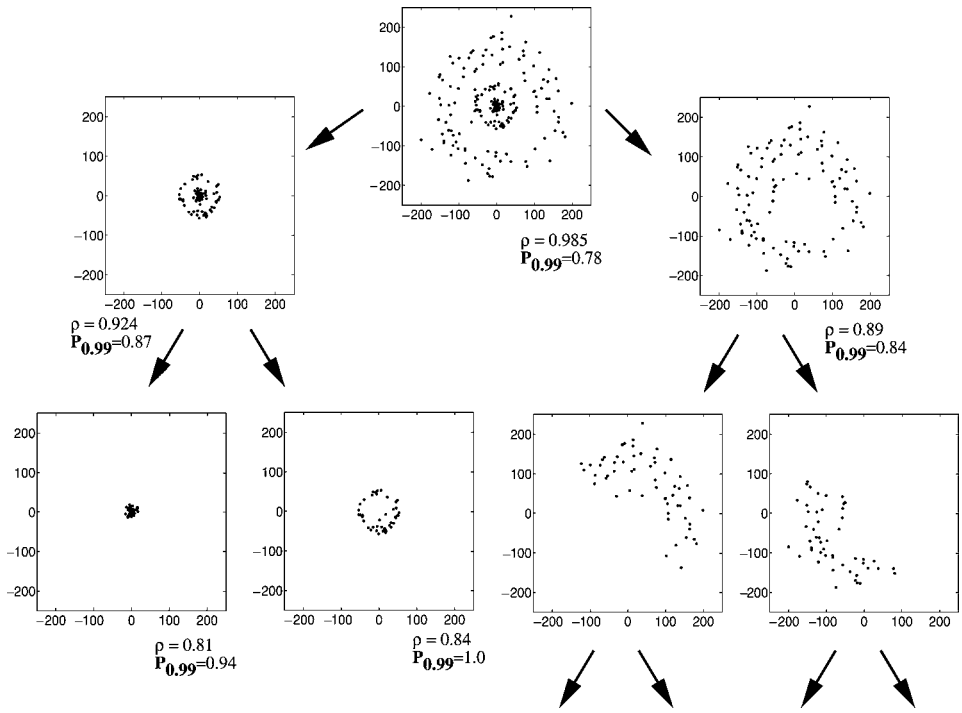

Figure 7. Cross-validated hierarchical clustering of three concentric rings.

Despite the success of the cross-validation method with respect to these splits, it fails to stop the partitioning of the outer ring. Here the null hypothesis of no structure is rejected, and the algorithm continued by splitting this ring into two halves.

The cross validation method proposed in this section for *pairwise* hierarchical clustering is inspired by Linial et al. (1997), who applied a similar method to hierarchical *vector* clustering. Both methods use internal indices of validation to stop the divisive process, and prune tree siblings that do not reflect meaningful structure. Other pruning techniques use external validation which is relative to some performance task. External validation combines supervised learning strategies with the unsupervised clustering problem, and they are applicable when a validation set of observations is available for which the correct classification is known. For example, the pruning technique in Fisher (1996) uses the partition tree to classify items from the validation set, while masking one of their attributes. Nodes in which all attributes can be reliably predicted are pruned.

4. Recovering structure in noisy data

One of the most attractive features of our algorithm is its ability to recover the structure of severely corrupted data. The reason for this robustness is that global information is taken

into account at every update step of the proximity matrix. In this section we consider three noise models, and investigate the stability of our results under controlled amounts of noise, for each of these models.

4.1. Proximity value noise

The first noise model considered is an "unreliable sensor", which is modeled by adding random noise to the proximity values. Specifically, each entry d_{ij} in the proximity matrix is perturbed by adding to it a Gaussian random variable e_{ij}, which has zero mean and its standard deviation equals αd_{ij}, α being the noise control parameter. In the case where d_{ij} becomes negative, it is set to zero.

Figure 8 shows a concrete example which we use throughout this section. We take 100 points from two $2D$ Gaussian distributions (50 points from each Gaussian). These points

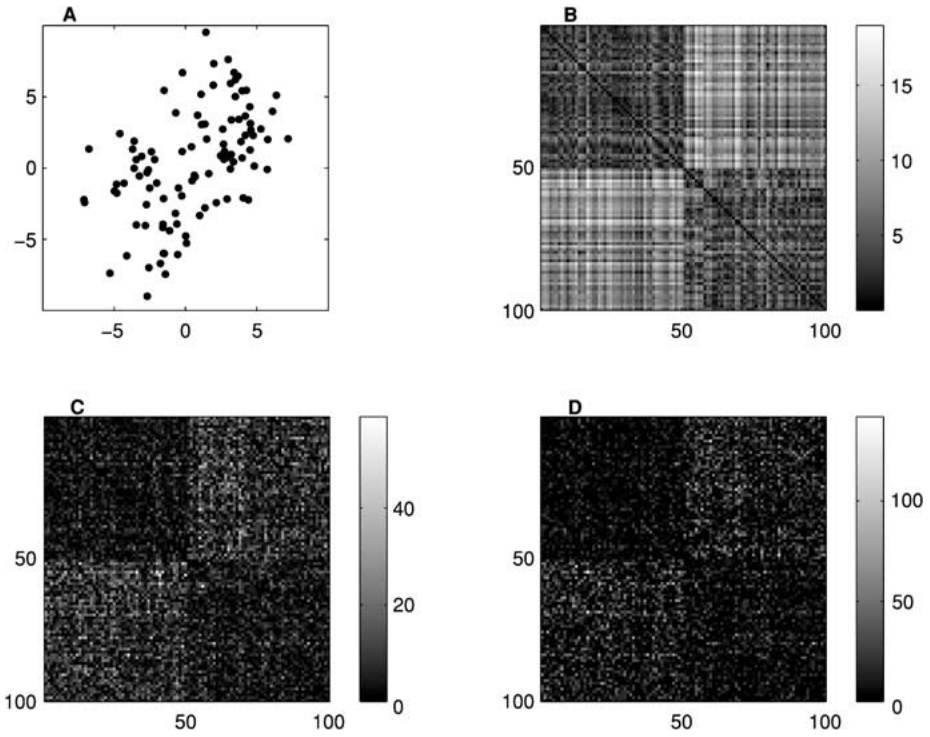

Figure 8. Investigating the noise robustness of the algorithm. (A) The data points from the two Gaussians. (B) The Euclidean pairwise distances between the data points. A and B are used throughout all the experiments described in this section. (C–D) Two examples of perturbed distance matrices from the Gaussian-noise model (Section 4.1) which were generated with $\alpha = 1.0$ and 2.5 (see text). Note that although C and D may look similar, their color-bars span over a different range.

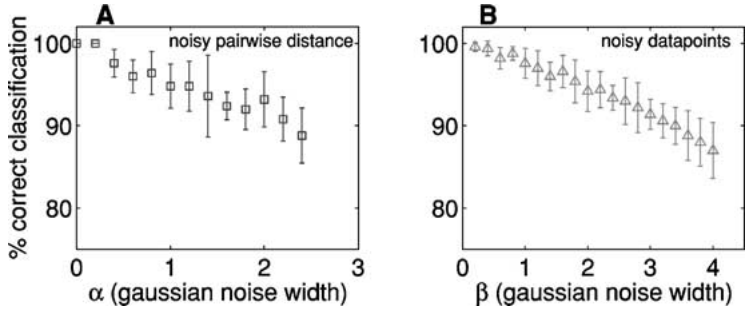

Figure 9. Algorithm performance for proximity noise and data point noise. (A) The averaged correct classification percentage for the data of figure 8, i.e. Gaussian noise added to each entry in the proximity matrix. Classification results are compared to the algorithm performance in the noiseless case, rather than to the 'true' classification of the data points. Error bars are the standard deviations of the classifications, computed from five independent experiments with the same value of the noise control parameter α (see text for noise model details). (B) Similar to A, the averaged correct classification is shown for the case of adding Gaussian noise (with control parameter β) to the position of each of the data points in figure 8(A) (see text for details). Error bars are standard deviations computed from five independent experiments with the same value of the noise control parameter β.

are shown in figure 8(A). The corresponding proximity matrix, where d_{ij} is the Euclidean distance between points i and j, is shown in figure 8(B). Two examples of perturbed matrices, which were obtained using $\alpha = 1.0$ and $\alpha = 2.5$, are shown (respectively) in parts C and D.

To quantify the effect of noise on the algorithm performance, we used a range of values of the control parameter, α. For each value, we generated five random matrices, and applied the clustering algorithm to each of them (with L_1 norm and *JS* divergence). Each clustering result is compared with the "un-noisy" clustering result obtained for the unperturbed matrix. The success rate is measured by the labeling agreement before and after the perturbation. Namely, if $l(\alpha)$ is the average number of points which remain with the same label, then the "average agreement" (or "success rate") $h(\alpha)$ is defined as $h(\alpha) = \max(\frac{l(\alpha)}{n}, \frac{n-l(\alpha)}{n})$. Note that the minimal agreement percentage is 50%.

The averaged agreement h as a function of α, for the data of figure 8, is shown in figure 9(A). We conclude that the error rate increases approximately linearly with α, and stays low even for a large perturbation of the data.

4.2. Data point noise

We also consider the case of noisy data points. In the current model we shift each of the data points shown in 8(A) according to a 2D Gaussian distribution with width β. Figure 9(B) shows the agreement between the classification of the noiseless data to the noisy data (similar to the evaluation in the previous section). The behavior is close to linear, with a slope that reflects that even for large noise values, which deform the original distribution of the data points, the algorithm performs rather well.

4.3. Missing data

There are at least two reasons for considering the case of partial proximity information (where some of the data in the proximity matrix are missing). One is that the measurement sensor might be malfunctioning, and occasionally does not supply proximity measurements for some pair of items. The other reason is that the proximity measurement might be too complex or time consuming (e.g. when a sophisticated and costly algorithm is applied for similarity analysis of images), or when the size of the problem is too large and one would like to avoid some of the proximity measurements.

The noise model which is suitable here is obtained by random elimination of entries in the proximity matrix. In order to have an appropriate control, we constrain the random elimination by forcing it to keep exactly k entries for every row and column, where k is a parameter. Since we assume that the original proximity matrix is symmetric, we perform a symmetric elimination (if d_{ij} is missing then d_{ji} is missing as well). Otherwise, the missing data could have been trivially recovered.

There are various ways to generalize our algorithm for this case. One of the simplest ways is to substitute the observed column average for the missing data in that column. This scheme is particularly helpful when the data is structured in clusters of similar sizes, since in this case the missing entries are typically assigned "neutral" values. Nevertheless, it is evident from figure 10(A) that even a symmetric problem becomes very hard when parameter k has a very small value.

The controlled parameter k, the number of remaining entries per column (in addition to the diagonal entry), is varied between 1 to 15. For each value of k, five random corrupted

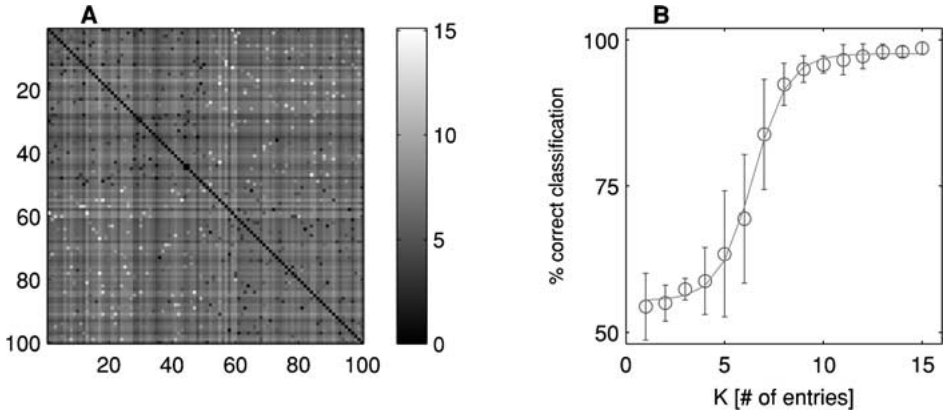

Figure 10. Overcoming missing data. (A) An example of a random corrupted matrix obtained for $k = 8$. At each row and column there are 8 original entries (plus the diagonal one), and the rest of the entries were set to the average value of the non-corrupted values of the column. Note that the vague structure that may be seen is due to the fact that the columns are ordered (like in the original matrix), such that the first 50 points belong to one cluster and the last 50 points belong to the other. However, under an arbitrary ordering no structure apparent by eye. (B) The average success rate $h(k)$ as a function of the number k of non corrupted entries per column. The average results for the example in A (and the other $k = 8$ matrices) was 92%. The step like curve is obtained by a least mean squares fit of a hyperbolic tangent function.

matrices were generated, and for each one of them the classification result was compared with the non-corrupted classification. The average agreement is shown in figure 10(B), where $h(k)$ is defined in the same way as $h(\alpha)$ above.

It is striking that even after losing 90% of the data, the algorithm results are hardly affected, and only a small fraction of the points is assigned to different clusters under this heavy corruption.

5. Experiments with some real data sets

In this section we consider three natural data sets. Two are taken from the UCI repository (Blake, Keogh, & Merz, 1998), and the third is a new data set of music pieces. Because of computational inefficiency considerations, for the clustering results reported here we turned off the "calibration" procedure of the cross-validation index (see Section 3) and we only report the resulting cross-validation indices obtained during the computations.

In Section 5.1 we consider the classical Iris data sets. Then, in Section 5.2 we consider the Isolet data set. An application to musical data is considered in Section 5.3.

5.1. The Iris data

This data set, attributed to Fisher (1936), is a classic (easy) benchmark in the pattern recognition literature. It is included in the UCI repository (Blake, Keogh, & Merz, 1998). The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The three classes are Iris Setosa (IS), Iris Versicolour (IVe), and Iris Virginica (IVi). Each data item is a 4-dimensional real vector representing four measurements of an Iris flower. Class IS is linearly separable from the other two (IVe and IVi), but IVe and IVi are not linearly separable. As data points are given as vectors, we computed the 150×150 pairwise distance matrix corresponding to the 150 data points before applying our algorithm. We used the Euclidean (L_2) distance between each pair of the given vectors (without any renormalization).

The clustering of the Iris data set by our algorithm is depicted in figure 11. The cross validation index (agreement percentage) is specified near each node (split). For example, the root node splits into two sets represented by the left node, containing 60 samples, and the right node, containing 90 samples. The cross-validation index for this split is $\rho = 0.98$ which means that there was a 98% agreement in the labeling of samples in the (random) cross-validation subset for this split. The leaves in this resulting dendrogram represent the three prominent clusters in the Iris data set corresponding to the three types of Iris flowers. In the figure we also show the (smaller) cross-validation indices of further splits below the leaves. We (arbitrarily) stopped the hierarchical clustering using the cross-validation index value of the first split. As can be seen the algorithm classified correctly 124 samples. This result is comparable with results obtained by other algorithms. For example, it is reported in Blatt, Wisemann, and Domany (1996) that their Super-Paramagnetic pairwise clustering algorithm classified correctly 125 samples in the data and that this result is the second best among eight other algorithms (the best performing algorithm on this example was the Minimum Spanning Tree).[9]

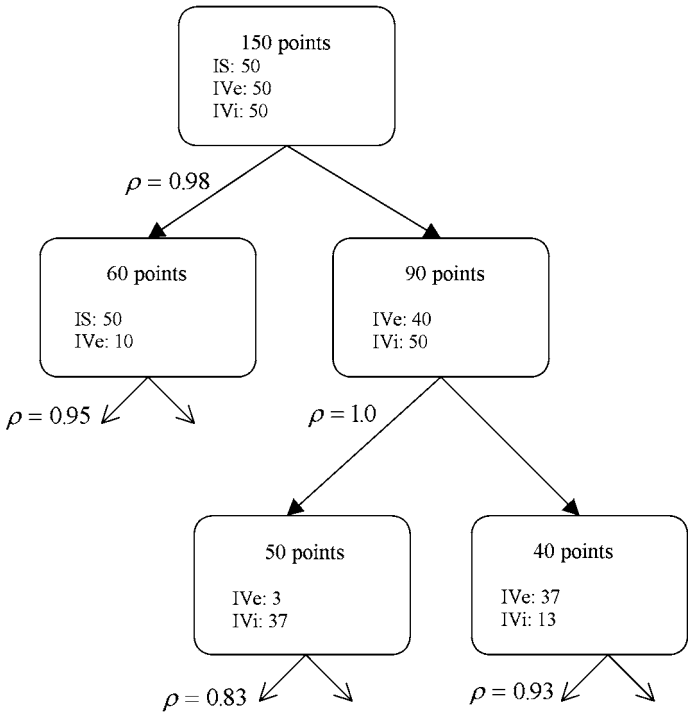

Figure 11. The hierarchy tree of the clustering of the Iris data set. Each node contains the number of data points in this cluster and number of elements from each of the Iris types in the cluster. The cross-validation index of each of the splits is shown near the appropriate node.

5.2. The isolet data

To test our algorithm on a larger and more difficult dataset we chose parts from a large speech dataset ISOLET of isolated-letter pronunciations created by Ron Cole (Fanty & Cole, 1991), publicly available at the UCI machine learning repository (Blake, Keogh, & Merz, 1998). The ISOLET dataset contains analysis vectors of different spoken names of English letters, spoken by subjects of both genders, different accents and varying English dialects. Each analysis vector contains 167 acoustic features that represent spectral coefficients, contour features, sonorant and post-sonorant features, etc. The exact description and ordering of the features is unavailable. All entries are real valued, scaled between -1 and 1.

In our experiment we chose to use only a limited subset of 8 letters B, D, P, T, M, N, I, A from the original dataset that contains all 26 letters of the English alphabet. The set of speakers in our experiment was derived from the fifth ISOLET dataset (isolet5) which contains 30 speakers. Each letter is pronounced twice, with one missing letter M sample, giving a total of 479 samples (feature vectors).

This clustering task attempts to find structure in the feature vectors data, related to the phonetical similarities apparent in the original spoken sources. Our choice of the letters was

guided by the principle that a clear phonetic distinction would exist between the different groups, and at most one articulatory difference would distinguish letters of the same group (Clark & Yallop, 1995). Thus, in the first group we have four letters with different consonant stops B, D, P, T at their beginnings and approximately similar articulatory features in their endings. Another group is M, N that is characterized by similar beginnings, and differences between the two nasal consonants at the letter pronunciation ending. The last pair A, I has two distinct vowels in the beginning and approximately similar ending.

Figure 12 shows a hierarchical clustering of the 479 samples, computed by our algorithm applied with L_2 norm and L_2 distance. The numbers at the cluster nodes of the hierarchy tree are the number of data points at each node (top), and the value in parenthesis is the cross validation index that allowed for a split of this cluster into two sub-clusters. In the leaf nodes of the tree we explicitly write the labels of the original letters that were found to belong to these leaf clusters, and whenever a leaf cluster significantly classifies a letter, we write this letter below the leaf.

It is evident that our clustering is partially successful. Some of the letters were identified and some were not. For instance, there is a separate leaf with no subdivision that corresponds to the nasal group M, N. The letter I was almost perfectly clustered. There is another partition that approximately distinguishes the letters A and I. In general, the clustering/separation among letters B, D, P, T was problematic.

These results are comparable with the clustering results of Blatt, Wisemann, and Donany (1996) who clustered the entire ISOLET data set. Although the paper (Blatt, Wisemann, & Donany, 1996) does not specify the errors (misclassifications) in their work, the N, M clustering is very similar, and the B, D, P, T separation is difficult as well (it occurs at a very late stage). It is interesting to note that the Blatt *et al*. clustering splits A and I into very far clusters, which can be considered as an undesirable effect that we did not encounter. (However, they used the complete set of 26 letters in the ISOLET data set and therefore, their task was more elaborate).

5.3. Music style classification

This new data set originates from midi files containing music scores from the 17th to 20th century (Schwob, 1998). Nineteen composers were selected, ranging from the Renaissance and the Baroque periods to the Romantic and Impressionist styles. The musical works were chosen according to a simple criterion of having three works by each composer, picking the shortest pieces among those appearing in the database.

The data set can be viewed, at a very rudimentary level, as a set of strings (musical voices) over an alphabet of pitches (the notes are represented as MIDI pitches). Several notes might appear simultaneously if the music contains chords (harmony) or polyphony (several voices). This is a temporal, multi-dimensional data set that is hard to analyze and cluster. The problem of clustering/classification is important for several music applications (Crawford, Iliopolous, & Raman, 1998; Rolland & Ganascia, 1999).

A pre-processing step was applied to the data for converting polyphonic music into a linear representation, and using intervals instead of pitch representation. Since our pairwise clustering algorithm operates on a distance matrix, an initial proximity matrix was calculated

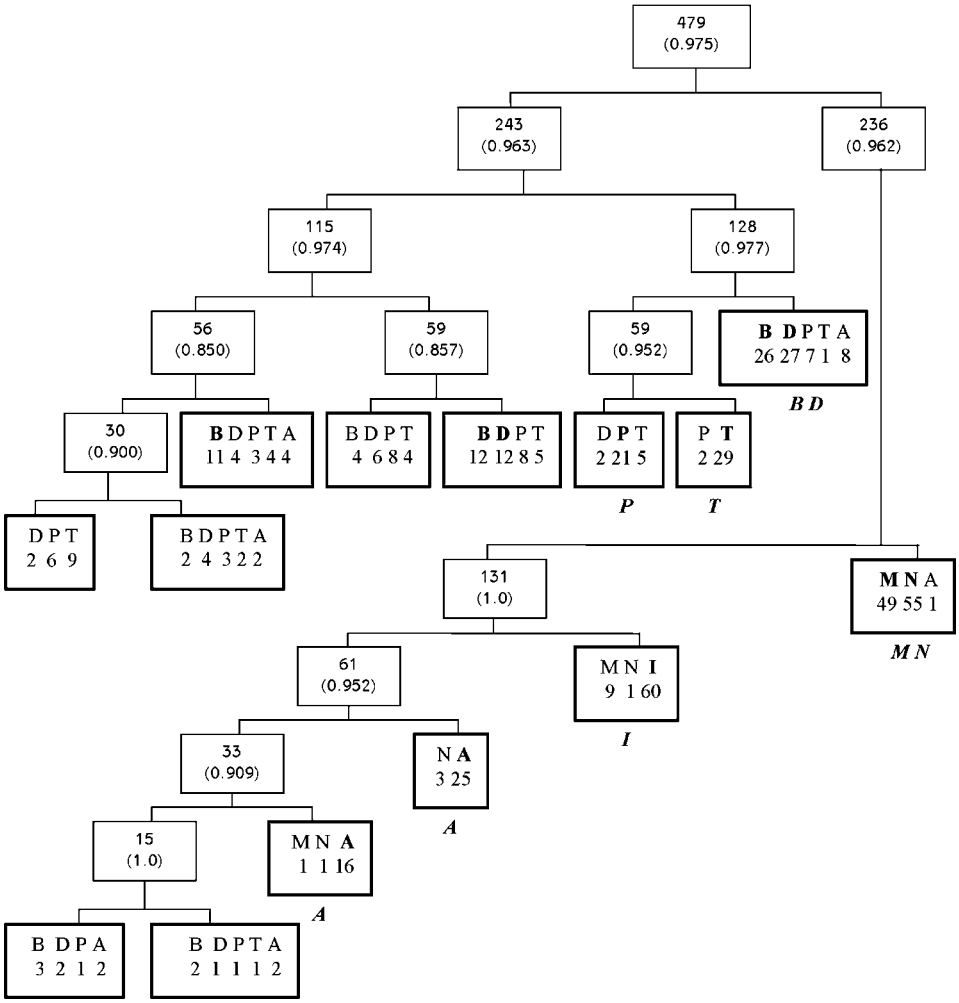

Figure 12. The hierarchy tree representing the clustering of the ISOLET data set. Each inner node of the tree represents a split and shows the number of data points in this cluster and the cross-validation split index (in parenthesis). The leaves of the tree are the resulting clusters, and hence contain the letters and their counts.

for every pair of music sequences. The dissimilarity between two sequences x and y was estimated using the JS-divergence between their underlying sources S_x and S_y, as discussed in El-Yaniv, Fine, and Tishby (1997).

It is interesting to note that the initial distance matrix, as shown in figure 13(A), is extremely noisy and almost no structure is apparent in the data set. Figure 13(B) shows the bipartition of the complete data set, given by the clustering. Black entries in the matrix correspond to distance zero, i.e. for coordinate (i, j) a black entry means a link between pieces number i and j. The pieces are ordered along the axes according to their chronological

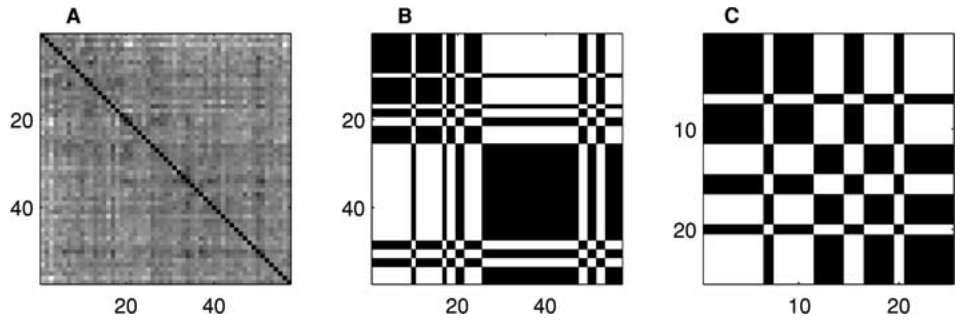

Figure 13. (A) Initial distance matrix obtained by calculating cross-entropy between midi sequences of works by 19 different composers, 3 works by each composer. (B) Clustering results of the distance matrix presented in A. The main separation of the two clusters occurs around examples 25–26, which corresponds to works by Beethoven. (C) Secondary sub-clustering of the first cluster in B (see text for details).

order, i.e. approximately corresponding to different stylistic periods. A list containing full names of the pieces of music appears in the Appendix. The music pieces separate into two main groups: the first cluster contains mostly works from early and classical periods, while the second cluster contains later romantic and impressionist works.

Figure 13(C) shows a secondary sub-clustering of the first cluster. The original piece numbers that correspond to the two sub-clusters (i.e. results of applying the clustering procedure to the works of music of the first cluster) are {1 2 3 4 5 6 8 9 11 12 16 18 24} and {7 13 14 15 19 22 23 25 48 49 52 53}. See the list of the pieces of music and their corresponding numbers in the appendix. The first sub-cluster contains mostly Renaissance and early the Baroque works, while the second sub-cluster contains mostly works of later Baroque and classical periods. Four works from the late 19th century appear in the second sub-cluster as well.

For further discussion of the results and description of the methods that were used to obtain the initial distance matrix, the reader is referred to Dubnov, El-Yaniv, and Assayag (1997).

6. Relation to other work

The clustering method presented in this paper shares features with a variety of other clustering algorithms. We discuss the similarities and differences with a few of these, in order to give the reader a more intuitive notion of the algorithm's dynamics and capabilities.

In very simple clustering problems the similarity values between clusters are much smaller than those within clusters. Hence there is a threshold θ, such that if all edges with a similarity weight smaller than θ are removed, the similarity graph is disconnected into components which represent the desired clusters.[10] Real problems are rarely that simple. The objective of clustering is usually more global, while edge thresholding is purely local. However, if the pairwise relations are replaced by higher order relations, then they start to reflect global properties and the simple connected components algorithm becomes surprisingly effective.

High order relations (which we call "collective" or "transitive" relations) have been defined in various ways. A simple and direct approach is taken in Smith (1993) and Gowda and Krishna (1979), where the dissimilarity value between nodes i and j is transformed to $m + n$ if node i is the mth nearest neighbor of node j, and node j is the nth nearest neighbor of node i. A much more complicated transformation is hidden in the min-cut algorithm for clustering (e.g., Wu & Leahy, 1993). It is known from the Gomory-Hu theorem (Gomory & Hu, 1961) that the minimal cut separates the nodes of the graph in such a way that the max-flow value for every two nodes within a component is larger than every max-flow value between components. Hence the min-cut algorithm is equivalent to the definition of max-flows as collective similarities, followed by thresholding. Furthermore, when a probability distribution function is imposed over all possible cuts (or data partitions), then a collective measure which is based on the pairing probabilities can be defined, where the pairing probability of every two nodes is their probability to be on the same side of a random cut. This higher order measure stands at the core of the methods in Blatt, Wisemann, and Domany (1996) and Gdalyahu, Weinshall, and Werman (1998).

The clustering method, which we propose in the current paper, differs from the above algorithms in one major way. In our method the transformation from pairwise to higher order relations is performed *iteratively*. We claim that the clustering problem which is obtained after one transformation step can be further handled using the same transformation tool.

The incremental improvement which we achieve using a recursion is demonstrated in the following comparison example, where we compare our method with a recently proposed spectral method, called "normalized cut". Spectral methods identify good partitions via the eigenvectors of the similarity matrix, or other matrices derived from it (Perona & Freeman, 1998; Shi & Malik, 1997; Kleinberg, 1998). A particular eigenvector v is treated as an indicator function: a threshold θ is selected, and each node i is assigned to one part if $v[i] > \theta$ and otherwise to the other part.

The normalized cut algorithm (Shi & Malik, 1997) can be presented as follows.[11] Given a symmetric similarity matrix M, define D to be a diagonal matrix with $D[i, i] = \sum_j M[i, j]$. The operation $D^{-1}M$ normalizes the sum of each row in the similarity matrix to one, hence a vector whose entries are all 1's is an eigenvector of $D^{-1}M$. Moreover, it is the principal eigenvector, with the largest eigenvalue (which is 1). The normalized cut algorithm uses the next principal eigenvector, denoted v_2, in order to partition the nodes into two parts; if $v_2[i] > \theta$ then node i is in one part, otherwise it is in the other.

On the other hand, our algorithm iterates the matrix $D^{-1}M$ by repeatedly computing new proximity indices from its normalized rows. Namely, we define element $[i, j]$ of matrix $M^{(t+1)}$ to be the JS-divergence between rows i and j of the matrix $[D^{-1}M]^{(t)}$. In figure 14 we observe that the indicator vector v_2 improves during the iterations, until it converges to a bi-valued vector. The data points which are used to generate this particular example are shown in figure 15, together with the partitions that are obtained from the iterated and the un-iterated vector v_2.

The idea of a recursive similarity manipulation is common to our work and that of McQuitty and Clark (1968), which was further analyzed in Kruskal (1978). The transition between pairwise similarities and high order, "collective", similarities is achieved in these works by computing statistical correlations between the matrix columns.

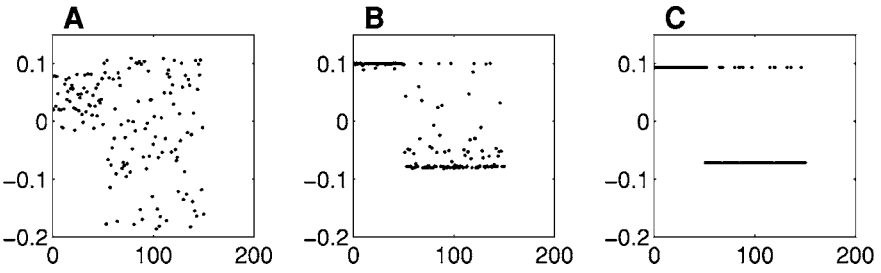

Figure 14. Evolution of the second eigenvector of $(D^{-1}M)^{(t)}$ during the iterations of our algorithm. Figures (A), (B) and (C) show the entries of the second eigenvector for $t = 0$, 6 and 28, respectively. The normalized cut algorithm (Shi & Malik, 1997) works by thresholding the values in A. The partition which is obtained is inferior with respect to the one obtained by thresholding the values in C (see figure 15). The points are ordered, hence the first 50 entries correspond to the inner cluster, and the last 100 entries correspond to the background.

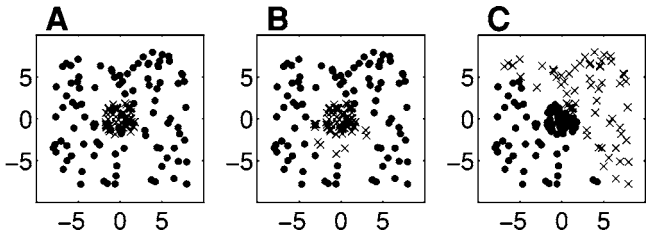

Figure 15. The effect of our iterative proximity transformation on the partition found by a recently proposed spectral method (Shi & Malik, 1997). (A) The data points. (B) The partition which is obtained by thresholding the iterated vector shown in figure 14(C). (C) Clustering by the normalized cut algorithm (Shi & Malik, 1997). The result is obtained by thresholding the values of the second eigenvector shown in figure 14(A).

More formally, construct the matrix $\hat{M}$ by subtracting from each column of the proximity matrix M its mean, and consider the operator $\mathcal{O}(M) = \hat{D}^{-\frac{1}{2}}\hat{M}^T\hat{M}\hat{D}^{-\frac{1}{2}}$, where $\hat{D} = diag(\hat{M}^T\hat{M})$. A sufficient condition is proved in Kruskal (1978) which guarantees that the t-folded composition $\mathcal{O}^t(M)$ converges when $t \to \infty$ to a bi-valued matrix M^* with either $+1$ or -1 entries. The fixed point matrix M^* is a permutation equivalent to a block diagonal matrix, with two diagonal blocks of sizes k and $n-k$ containing the $+1$ entries, and two off diagonal blocks containing the -1 entries. This indicates a partition of the data into two clusters. The sufficient condition which is shown in Kruskal (1978) to guarantee the convergence, is that $\max |M[i, j] - M^*[i, j]| < (-1 + \sqrt{1+4e^2})/2e$, where $e = 8(k/n)(1-k/n)$.

Using statistical correlations as an intercolumnar similarity measure is similar to our L_2 distance experiment. In this paper we also extend and use the information theoretic measure, the JS-divergence. A similar information theoretic measure (the Kullback Leibler divergence) is used by other clustering algorithms. For example, Pereira, Tishby and Lee (1993) use the KL-Divergence in a different context, acting on vector spaces and not on similarity matrices. More recent works (Tishby, Pereira, & Bialek, 1999; Slonim & Tishby, 2000) deal with clustering through extracting the information that the resulting clusters retain.

7. Conclusions and future work

In this paper we presented a novel approach to pairwise clustering. Our basic bipartition clustering algorithm it iteratively employs a "collective" proximity measure to improve and sharpen the proximity matrix, until it converges to a clear partition.

This algorithm was found to yield good clustering results for several examples and it appears to be comparable, in terms of its clustering quality, to other known algorithms. Moreover, our experiments indicate that our algorithm is extremely robust and can recover the structure for very noisy data sets and incomplete ones.

We have also presented a new cross-validation technique. The proposed technique is general and may work with other bipartition algorithms (that use vectorial and/or pairwise data representations). Our preliminary experiments with hierarchical clustering using this technique indicate that this technique is viable.

Our basic bipartition algorithm, which is essentially nonparametric enjoys other attractive features. For example, we can equally use a similarity measure instead of a dissimilarity measure in our two-step transformation. This is a non-trivial feature as this flexibility is not always supported by other methods. For example, the graph based algorithms whose cost function is related to cut capacity (Wu & Leahy, 1993; Shi & Malik, 1997; Blatt, Wisemann, & Domany, 1996; Gdalyahu, Weinshall, & Werman, 1998) are all formulated in terms of similarity values. This is essential, since their cost optimization problem is related to the min-cut problem of graphs, which can be solved in polynomial time. Converting these methods to dissimilarity graphs results in attempts to solve max-cut related problems, which are NP-hard.

The proposed clustering algorithm has various deficiencies. Perhaps the main drawback of the algorithm is its intense time complexity. Each iteration of our bipartition algorithm (re-normalization and re-estimation steps) takes $\Omega(n^3)$ where n is the size of the data set. We have experimented with several approximation methods to reduce the time complexity. One of the more successful methods is to project the probability vectors, which represent the data set points, onto some random set of components. This way we can reduce the time complexity to $O(n^2)$. We should also mention that our algorithm is highly parallelizable. For example, in the extreme case of $O(n^2)$ parallel processing units, each iteration can be completed in $O(\log n)$.

Another difficulty we have encountered was a bias of the algorithm to partition the data into clusters of similar size. If the number of data points from one source is very different form the other source, this may result in an undesirable partition. However, the specific results depend on the distance between the clusters relative to the inter-cluster distances.

We leave several directions for future work. First, it would be very interesting to prove or disprove our conjecture about the convergence of the bipartition algorithm. In particular, it is important to characterize the set of norms and proximity measures that allow for convergence. Preliminary results suggest that global normalization schemes on the proximity matrix, (instead of the column normalization scheme that we have presented here) should also be investigated.

Second, it would be interesting to examine more closely our cross-validation technique and test it with various data sets. It may also be possible to analytically characterize the

distribution of the cross-validation indices with respect to random matrices. This possibility is particularly attractive because it will eliminate the need for the costly Monte-Carlo sampling that we used here.

Finally, we believe that it is possible to extend our method to ordinal data representations, where the proximity information is qualitative and not quantitative. A possible direction was pointed out in Section 6, where the dissimilarity transformation used by Smith (1993) was mentioned as a possible candidate.

Appendix: The music style data set

The following table contains the names of the clustered pieces of music. Due to space limitations, we do not specify here the full names of the pieces but rather the shortcuts that correspond to the midi file names, as they appear in Schwob (1998).

Eyck	Bach	Beethoven	Liszt	Debussy
courant = 1	cantata = 13	baga1 = 25	canzone = 37	clardlun = 49
daphne = 2	chorale = 14	baga2 = 26	liebstrm = 38	deb_ara1 = 50
denavond = 3	concerto = 15	baga3 = 27	valseoub = 39	debptng = 51
Dowland	Handel	Brahms	Tchaikovsky	Ravel
fantasia = 4	gfhair = 16	brahm117 = 28	nut0over = 40	ravelpav = 52
lacrimae = 5	gfhgav = 17	brahm118 = 29	nut1mrch = 41	ravelson = 53
queen_el = 6	gfhouv = 18	brahmiz3 = 30	swanlake = 42	ravgleg = 54
Purcell	Haydn	Grieg	Mussorgsky	Satie
pur_borr = 7	gdance1 = 19	gr_ariet = 31	pm1gnome = 43	gnoss = 55
purmarch = 8	gdance5 = 20	gr_nocte = 32	pmcastle = 44	gymno = 56
purmfaw = 9	gdance6 = 21	gr_waltz = 33	prmchick = 45	gymnop02 = 57
Couperin	Mozart	Chopin	Scriabin	
barimyst = 10	fnt_dmin = 22	chpn_p7 = 34	scriab01 = 46	
coumoiss = 11	minuet6d = 23	chpre1 = 35	scriab06 = 47	
cpf_bird = 12	mozcap = 24	opus28n4 = 36	scriab14 = 48	

Acknowledgments

We would like to thank Prof. Dr. Hans-Hermann Bock for sending us the papers of Kruskal and of McQuitty and Clark. This work was partially supported by a grant from the Ministry of Science, Israel. R. El-Yaniv is a Marcella S. Geltman Academic Lecturer.

Notes

1. While preparing the manuscript for publication, we have learned of a similar (although less general) algorithm presented by McQuitty and Clark (1968), in which the correlation between data points is iteratively calculated, based on the correlation matrix columns.
2. Clearly, the nature of the proximity matrix will play a crucial role in the success of any clustering algorithm. While for some sets of data points, there is a natural distance measure, in other cases, finding the 'right' measure is problematic. An example for a natural distance is given in figure 1(A) and (B) where we deal with a set of points in a 2D plane, and L_2 was used as the distance measure between two points.

3. Throughout the paper we regard dist(·) as a dissimilarity measure, but our method can also accommodate similarity measures.
4. Due to the nature of the JS-divergence, the values of the matrix are bounded between 0 and 1. In the current example, the iterative process was stopped when none of the matrix values changed by more than 10^{-5}. The matrix entries in the final iteration are 0 and 1 within this resolution.
5. A classification error occurs when a point is assigned to a cluster in which the majority of the points originated from the other source.
6. It is feasible to obtain a full solution and carry out a stability analysis for the simple cases like $n = 3$ (using the L_2 norm and the L_2 distance).
7. We have also experimented with a large set of random matrices of different sizes. All examples we have tested converged within a few dozens to a few hundreds of iterations.
8. As noted earlier, ρ values are averaged over the cross-validation index of independent runs. For example, in this case the index 0.696 was an average of the three independent indices 0.529, 0.911, and 0.648.
9. Note that the transformation used by Blatt, Wisemann, and Domany (1996) to generate the pairwise matrix from the vectorial representation was different than ours. Applying a transformation similar to theirs, we have achieved significantly better results.
10. The generalization to k-connected components or cliques is computationally less attractive.
11. This presentation of the algorithm differs from the original presentation given by Shi and Malik (1997), but it can be shown to be equivalent. The original formulation uses the Laplacian matrix of the graph, and is motivated by a continuous approximation to a certain discrete partition problem.

References

Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford: Oxford Press.

Blake, C., Keogh, E., & Merz, C. (1998). UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Blatt, M., Wisemann, S., & Domany, E. (1996). Clustering data through an analogy to the Potts model. In D. Touretzky, M. Mozer, & M. Hasselmo (Eds.), *Advances in neural information processing system* (Vol. 8, pp. 416–422), Cambridge, MA: The MIT Press.

Buhmann, J., & Hofmann, T. (1995). Pairwise data clustering by deterministic annealing. Technical Report IAI-TR-95-7, Department of Computer Science, University of Bonn. ftp://ftp.informatik.uni-bonn.de/pub/paper/infIII/IAI-TR-95-7.ps.gz.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. In *Proceedings of the Fifth International Machine Learing Conference* (pp. 54–64). Ann Arbor MI: Morgan Kaufmann.

Clark, J., & Yallop, C. (1995). *An introduction to phonetics and phonology (Blackwell textbooks in linguistics, Vol. 9)*. Oxford: Blackwell Pub.

Crawford, T., Iliopolous, C., & Raman, R. (1998). String matching techniques for musical similarity and melodic recognition. *Computing in Muiscology*, *11*, 73–100.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, 39* (Series B), 1–38.

Dubnov, S., El-Yaniv, R., & Assayag, G. (1997). Universal classification applied to musical sequences. In *Proceedings of the International Computer Music Conference*. Ann Arbor, Michigan.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*. New York: Wiley.

El-Yaniv, R., Fine, S., & Tishby, N. (1997). Agnostic classification of markovian sequences. In M. Jordan, M. Kearns, & S. Solla (Eds.), *Advances in neural information processing system* (Vol. 10). Cambridge, MA: The MIT Press.

Fanty, M., & Cole, R. (1991). Spoken letter recognition. In R. Lippmann, J. Moody, D. Touretzky, & S. Hanson (Eds.), *Advances in neural information processing system* (Vol. 3., pp. 220–226). Cambridge, MA: The MIT Press.

Fisher, D. (1996). Iterative optimization and simplification of hierarchical clusterings. *J. Artifi. Intel. Res., 4*, 147–179.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7:2*, 179–188.

Gdalyahu, Y., Weinshall, D., & Werman, M. (1988). A randomized algorithm for pairwise clustering. In M. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems* (Vol. 11). Cambridge, MA: The MIT Press.

Gomory, R., & Hu, T. (1961). Multi-terminal network flows. *SIAM Journal of Applied Mathematics, 9*, 551–570.

Gowda, K.C., & Krishna, G. (1979). The condensed nearest neighbor rule using the concept of mutual nearest neighborhood. *IEEE Transactions on Information Theory, 25*, 488–490.

Gutman, M. (1989). Asymptotically optimal classification for multiple tests with empirically obsereved statistics. *IEEE Transactions on Information Theory, 35:2*, 401–408.

Jain, A., & Dubes, R. (1988). *Algorithms for clustering data.* New Jersey: Prentice-Hall.

Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*.

Kruskal, J. (1978). A theorem about CONCOR. Technical Report MH 2C-571, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974.

Kullback, S. (1959). *Information theory and statistics.* New York: Wiley & Sons.

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory, 37:1*, 145–151.

Linial, M., Linial, N., Tishby, N., & Yona, G. (1997). Global organization of protein segments using new algorithms for metric empbedding and clustering. *Journal of Molecular Biology, 268*, 539–556.

McQuitty, L., & Clark, J. (1968). Clusters from iterative intercolumnar correlational analysis. *Educ. Psych. Meas., 28*, 211–238.

Michalski, R., & Stepp, R. (1983). Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligenc, 5*, 219–243.

Pereira, F., Tishby, N., & Lee, L. (1993). Distributional clustering of English words. In *Proc. of the 31st Annual Meeting of the Association for Computational Linguistics*.

Perona, P., & Freeman, W.T. (1998). A factorization approach to grouping. In *Proceedings of ECCV.*

Rolland, P., & Ganascia, J. (1999). Musical pattern extraction and similarity assesment. In E. Miranda (Ed.), *Reading in music and AI*. New York: Harwood Academic Press.

Rose, K., Gurewitz, E., & Fox, G. (1992). Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory, 38*.

Schreibman, A. (2000). Stochastic modeling for efficient computation of information theoretic quantities. Master's thesis, Hebrew University.

Schwob, R. (1998). The classical midi archive. http://www.prs.net/midi.html.

Shi, J., & Malik, J. (1997). Normalized cuts and image segmentation. In *Proceedings of CVPR.*

Slonim, N., & Tishby, N. (2000). Data clustering by Markovian relaxation and the information bottleneck method. *Advances in Neural Information Processing Systems* (Vol. 13, pp. 640–646). Cambridge, MA: The MIT Press.

Smith, S. (1993). Threshold validity for mutual neighborhood clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15*, 89–92.

Tishby, N., Pereira, F., & Bialek, W. (1999). The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing* (pp. 368–377).

Wallace, C., & Dowe, D. (1994). Intrinsic classification by MML—the SNOB program. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence* (pp. 37–44). UNE, Armidale, NSW, Australia: World Scientific.

Wong, Y. (1993). Clustering data by melting. *Neural Computation, 5*, 89–104.

Wu, Z., & Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15*, 1101–1113.

Received February 15, 1999
Revised August 1, 2000
Accepted December 1, 2000
Final manuscript May 8, 2001